ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.1, January 2007, pp.85−93
DOI: 10.1360/jos180085

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于目标距离评估的启发式 Web Services 组合算法*

温嘉佳+, 陈俊亮, 彭 泳

(网络与交换国家重点实验室(北京邮电大学),北京 100876)

A Method of Heuristic Web Services Composition Based on Goal Distance Estimate

WEN Jia-Jia+, CHEN Jun-Liang, PENG Yong

(State Key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications), Beijing 100876, China)

+ Corresponding author: Phn: +86-10-62281143, E-mail: wenjiajia@263.net

Wen JJ, Chen JL, Peng Y. A method of heuristic Web services composition based on goal distance estimate. *Journal of Software*, 2007,18(1):85−93. http://www.jos.org.cn/1000-9825/18/85.htm

Abstract: With the growing number of available Web Services, it is an imperative task to study how to compose Web Services based on the requirements of the customers, which is the main focus of this paper. The proposed method automatically composes Web Services directly according to the customers' requirements, and then executes the composed services to achieve the customers' goals. Based on the historic records of Web Services compositions, this method uses a heuristic approach to adjust the composition scheme. Experimental results show that the method is well fits for the volatile environment and yields better performance over other algorithms.
Key words: Web service; semantic; service composition; heuristic

摘 要: 随着可用 Web Services 数量的快速增长,如何根据用户的需求来自动组合 Web Services,生成满足用户需求的组合业务,成为一项亟待解决的课题.提出了一种基于用户需求目标距离评估的启发式算法,通过该算法,动态调用 Web Services 来自动生成满足用户所需目标的 Web Service 组合,同时,该算法还能够根据 Web Services 组合经验,对以后的 Web Services 组合方案进行调整.实验结果表明:该算法能够很好地适应网络上 Web Services 的不稳定情况,与同类算法进行性能比较,也显示出基于目标距离评估的算法具有较好的性能.
关键词: Web 服务;语义;服务组合;启发式
中图法分类号: TP311 **文献标识码**: A

随着近年来 Internet 的发展,Web Services 作为一种新技术广受关注.Web Services 中的接口定义语言 WSDL[1]和内容传输格式 SOAP[2]已经成为 W3C 的草案和建议标准,然而,人们需要的往往不是单一的、简单的 Web Services,而是 Web Services 组合出来的新业务.目前,工业界采用的 Web Services 组合方式都是基于手工生成的工作流模式,比如 IBM、微软等公司推出的 BPEL4WS[3]技术以及惠普公司提出的 EFlow[4]技术.这些技术的缺点在于需要专业的人员手工生成业务,而不能根据用户的需求直接通过机器生成业务.其原因在于,目前的 Web Services 的标准 WSDL(Web service definition language),UDDI(universal description, discovery, and

* Supported by the National Natural Science Foundation of China under Grant No.60432010 (国家自然科学基金)
Received 2005-07-08; Accepted 2006-04-10

integration),SOAP(simple object access protocol)都不具备机器可以理解的语义信息.

为了使机器能够理解并可以自动地处理信息,人们进行了以 RDF[5],OWL[6]为基础的下一代 Web 的研究,Tim Berners-Lee 称其为“Semantic Web”[7].而将 Semantic Web 技术和 Web Services 技术加以结合,产生的 Semantic Web Services 技术则可以使机器对 Web Services 进行自动化的组合,利用已有的一些人工智能或者程序验证的方法,从用户的需求直接生成新业务.这种从需求直接自动生成 Web Sevices 组合业务的方式已成为目前研究的热点.

本文提出了一种新的 Web Services 的自动组合和执行算法.这种算法的新颖之处在于:整个 Web Services 的组合和调用是联系在一起的,算法能够根据已有的 Web Services 的成功或失败经验动态地进行组合,而不是将组合出来的业务绑定在具体的某个或某些 Web Services 上.如果因为网络原因导致 Web Services 调用失败,或者因为其他原因造成 Web Services 不符合用户的需求,算法并不会终止,而是尽最大可能地寻找其他符合条件的 Web Services.

Web Services 在网络上是不断增加和变动的.由于网络的不可靠性和不同 Web Services 提供商的 QoS、服务价格等因素的不同,Web Services 的组合不能绑定在固定的某些 Web Services 上.而对于某个 Web Services 能否满足用户的需求,是不能事先决定的,只有在用户调用以后,系统对 Web Services 的调用结果根据用户提供的需求约束进行判断,才能够了解.

这种算法的核心思想是将 Web Services 的组合问题看作是一个状态空间的搜索问题.本文提出一种启发式的方法,该方法能够有效地利用已有的 Web Services 组合经验,从而在大多数情况下都能够高效地完成状态空间的搜索.

本文第 1 节定义一些基本概念,并对算法的基本思想加以阐述.第 2 节详细介绍算法的核心——基于目标距离评估的启发式计算方法.第 3 节介绍算法实现以及对算法的应用举例.第 4 节介绍国内外相关的研究.最后是总结和下一步的工作.

1 基本思想

Web Services 的组合过程,实际上是一系列用户目标的实现过程,比如:用户的目标是去外地旅行(travel).为了实现这个目标,中间可能需要实现一些必需的其他目标,比如:为了到达这个旅游景点(travel site),需要了解这个旅游景点在哪个目标城市(destination city);为了到达这个城市,需要购买飞机票(ticket).用户可以针对这些目标提出一些约束,比如要求机票价格低于 1 000 元等.而用户也必须提供一些基本的信息,比如他(她)的出发日期(date)、出发城市(source city)和要去的旅游景点.图 1 显示了这些 Web Service 与目标之间的关系.

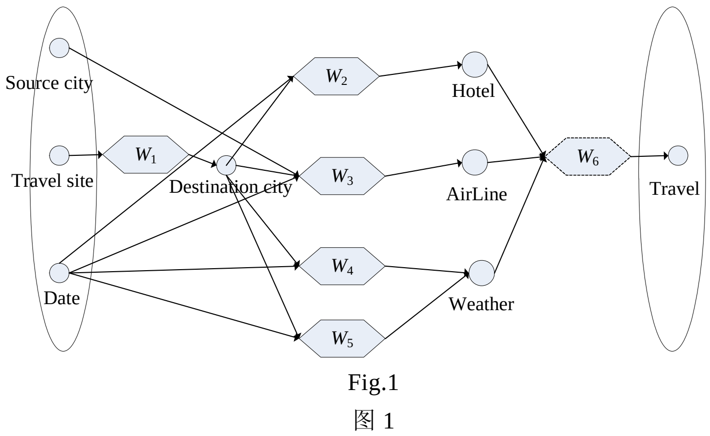


Fig.1

图 1

我们定义一些基本概念如下:

本体概念集合 G:Web 服务组合的所有本体的集合.比如旅游领域的本体:机票、宾馆(hotel)、出发城市(source city)、目标城市.这些都是由领域专家定义的本体库,Web 服务通过这些本体库定义其 OWL-S[8]语义描

述.图 1 中小圆框表示的就是一个个本体概念.

目标集合 S_g:该集合定义用户的需求,包含若干个本体概念.比如在旅游领域里面,用户可能需要一个机票信息(airline ticket)、租赁一辆出租车(taxi).显然,目标集合 $S_g \subset G$,如图 1 中的实线椭圆框所示.

初始条件集合 S_0:该集合定义用户能够提供给系统的初始条件,包含若干个本体概念.比如日期、出发城市、目标城市.显然,初始条件集合 $S_0 \subset G$,如图 1 中的虚线椭圆框所示.

Web Services 库 W:W 是 Web Services 的集合.对于 $W_i \in W$ 的任何一个 Web Service,都可以定义如下:W_i 为一个三元组 $W_i:=\langle Input(W_i),Output(W_i),Exception(W_i)\rangle$,其中:$Input(W_i) \subset G$,表示一个 Web Service W_i 需要的输入参数;$Output(W_i) \subset G$,表示 Web Service W_i 在存在 $Input(W_i)$的基础上,调用成功后输出的参数;$Exception(W_i)$是 Web Service 可能抛出的异常的集合.图 1 中六边形框图表示的就是一个个 Web Services.

比如一个获取机票信息的 Web Service,其输入条件可能是出发城市、目标城市和日期,其得到的输出是与机票相关的信息,而异常状况可能是在这两个城市之间不存在航班,所以,定义这个 Web Service W_i 为$\{Input(W_i), Output(W_i),Exception(W_i)\}$, 其中 :$Input(W_i)=\{Source\ City,Destination\ City,Date\}$,$Output(W_i)=\{Airline\ Ticket\}$, $Exception(w)=\{NoAirLine\ Exception\}$.

另外,定义虚 Web Service 的概念.所谓虚 Web Service,实际上是领域专家写的 Web Service,表示目标和目标之间的分解关系.虚 Web Service 也是一种 Web Service,但虚 Web Service 不是一个网络上实际存在的 Web Service.虚 Web Service 的调用永远成功,不会抛出异常.

比如目标旅行,这个目标的达到依赖于 3 个目标的达到,包括宾馆(hotel)、机票(airline ticket)和天气(weather),而旅行这个目标不会是任何一个 Web Service 的输出目标.事实上,旅行是一个高层的目标,这个目标可以分解成宾馆、天气和机票 3 个目标,这就需要领域专家定义一个虚 Web Service w,$Input(w)=\{Hotel,Weather, Ticket\}$,$Output(w)=\{Travel\}$,$Exception(w)=\varnothing$.本文中的算法将虚 Web Service 等同于其他 Web Service,通过这个虚 Web Service,实现将目标旅行分解成目标宾馆、天气和机票,只不过不调用网络上实际的任何东西而已.虚 Web Service 的思想来源于 HTN[9,10]技术中的任务分解机制.图 1 中的虚线六边形 $W6$ 表示的就是一个虚 Web Service.

约束条件 C:约束条件是以目标为常量的一阶谓词逻辑,表示用户对于目标实现的条件限制,比如 $Lessthan(Price(ticket),1000)$.

这样,本文将 Web Service 的组合问题看作一个五元组$\langle G,S_0,S_g,W,C\rangle$,而 Web Services 组合问题的实现看作是一系列目标的获取过程$\langle S_0,S_1,S_2,\ldots,S_n\rangle$,其中,$S_g \subset S_n$,即通过 Web Services 的调用,最终获得用户需要的目标 S_g.本文定义相应的 Web Services 调用序列$\langle W_0,W_1,W_2,\ldots,W_{n-1}\rangle$为 Web Services 组合问题的解决.

对于一个 Web Service W_i,如果 $Input(W_i) \subset S_i$,那么称 W_i 为在当前状态下可行的 Web Service,如果调用了 W_i,其结果不是 $Exception(W_i)$,且其 $Output(W_i)$满足约束条件 C,那么更多的目标将被获取,$S_{i+1}:=Output(W_i) \cup S_i$.

2 核心算法

在 Web Service 的组合过程中,最简便的方式是采用遍历的方式,对当前可行的 Web Services 进行逐个调用,以期能够得到预期的 S_g.但是毫无疑问,这种方式是一种低效的方式.本文提出的算法针对当前状态下可行的 Web Service,对于这些 Web Service 进行评估,判断其与目标 S_g 的距离,然后调用目标距离最小的 Web Service,从而高效地实现 Web Services 的组合.

算法框图如图 2 所示,首先需要定义两个基本概念:目标距离因子和目标距离.

下面说明目标距离因子的计算,Web Services 的目标距离因子可以看作是 Web Services 执行的一个代价.Web Services 的目标距离因子δ根据以下的公式计算:

对于所有 Web Services W,目标距离因子$\delta_0(W)$的初始值为 1;

当这个 Web Service 第 i 次调用以后,如果第 i+1 次调用成功,

$$\delta_{i+1}(W)=\delta_i(W)-1/2^{Success(W)}.$$

而如果第 i+1 次这个 Web Service 的调用抛出异常或者是不符合用户的约束条件 C,那么,

$$\delta_{i+1}(W)=\delta_i(W)+1/2^{Fail(W)},$$

其中:$Success(W)$为这个 Web Service 的调用成功总次数;$Fail(W)$为这个 Web Service 的调用失败总次数.

由以上目标距离因子的计算方法可以看出:若一个 Web Service 调用成功的次数越多,则其目标距离因子越小;若一个 Web Service 调用失败的次数越多,则其目标距离因子越大.算法认为:一个 Web Service 执行成功一次和执行失败一次,其差距是显然的.而一个 Web Service 执行成功/失败 100 次和 101 次,其差距则并不明显.因此,算法定义的公式在连续成功/失败后,对目标距离因子的调整量呈指数方式衰减.显然,0<δ<2.

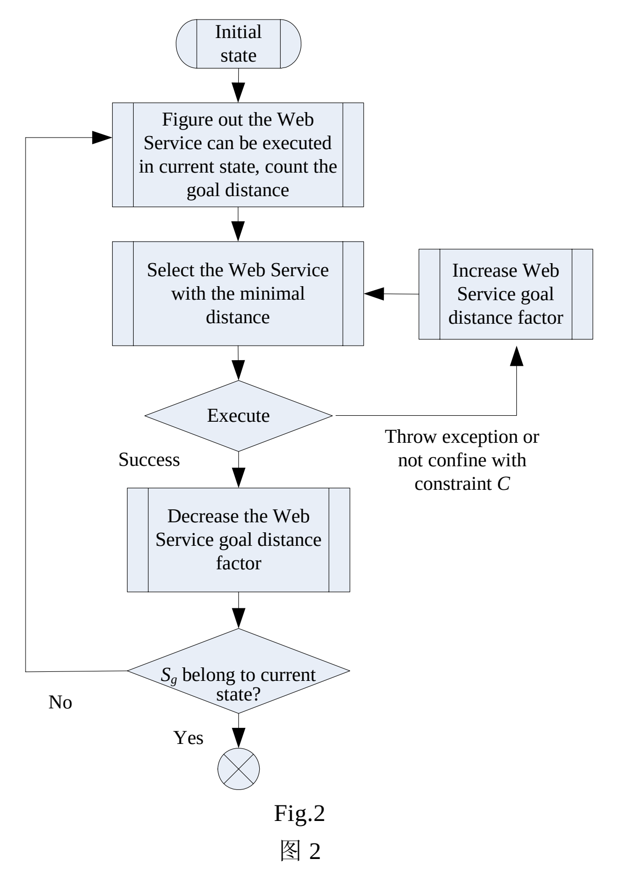


Fig.2

图 2

注意到,在图 2 中,无论是在 Web Service 调用成功、抛出异常或者是在这个 Web Service 的结果不符合约束条件的时候,算法都将相应地调节 Web Service 的目标距离因子.

定义一个 Web Service 的目标距离 $D(W_0)$为对于一个 Web Services 组合问题的解决$\langle W_0,W_1,W_2,\ldots,W_{n-1}\rangle$,

$$D(W_0)=\sum_{k=0}^{n-1}\delta(W_k)\ .$$

对于相应的 Web Services 组合的实现$\langle S_0,S_1,S_2,\ldots,S_n\rangle$,定义目标 g 的目标距离为

$$D(g)=\sum_{k=m}^{n-1}\delta(W_k)\ ,$$

其中,$g\in S_{m-1}$.如果 $g\in S_n$,显然,$D(g)=0$.

一个 Web Service 组合问题可能出现不同的解决办法.优选目标距离最小的 Web Service,这是本文算法的基础.

实际的目标距离在 Web Services 组合问题的解决不知道的情况下是无法计算的,但是,可以利用启发式的

方法对其进行估算.

计算当前状态下可行的 Web Services 集合与其目标距离的算法伪代码如图 3 所示.

```
procedure RegressionEstimate
Input:All of the Goal Set G, Parameter Set S_i, User Goal Set S_g, Web Service Set W,
      User Constraint C, the Limit of Composed Web Service Number: Limit
Output:Web Service Set W_pp, for all W_i∈W_app, the Web ServiceW_i can be executed in
       current state, all the W_i are with a goal distance estimation D(W_i)
01 begin
02 TTL=0;W_app=∅
03 forall (g_i∈G)
04     if g_i∈S_g then D(g_i)=0
05                 else D(g_i)=∞
05 forall (W_i∈W)
06     D(W_i)=∞
07 while (TTL<=Limit && S_g!=∅)
08   tempG=∅;W_g=∅
09   forall (g_i∈S_g)
10       forall (Output(W_i) has g_i && D(W_i)>D(g_i))
11           D(W_i)=min(δ(W_i)+D(g_i),D(W_i))
12           if (Input(W_i)⊂S_i)
13             W_app=W_app∪W_i
14           forall (g_k⊂Input(W_i))
15               D(g_k)=min(D(W_i),D(g_k))
16               tempG=tempG∪g_k/S_i
17   S_g=tempG
18   TTL++;
19 end
```

Fig.3

图 3

在本文中,启发式目标估算的方法是从目标 S_g 开始对当前状态 S_i 进行回溯,首先得到所有能够实现这些目标的 Web Services,而这些 Web Services 的调用依赖于一定的前提 $Input(W)$.以这些前提 $Input(W)$作为新的目标,再次进行回溯,直到所有前提都实现为止.

由于算法执行的过程中可能出现回路,所以,算法定义了一个最大值 *Limit*,如果循环的次数超过 *Limit*,则启发式目标估算算法结束.

算法开始时,凡是属于 S_g 的目标 g,其目标距离为 0;否则,其目标距离为无穷大.而对于所有的 Web Services,算法初始值都认为其目标距离为无穷大.

对于回溯过程中的每一个 Web Service W_i.算法估计其距离为 $D(W_i)=\min(\delta(W_i)+D(g_i),D(W_i))$,其中,$g_i\subset Output(W_i)$.

对于回溯过程中每一个目标 g_k,算法估计其目标距离为 $D(g_k)=\min(D(W_i),D(g_k))$,其中,$g_k\subset Input(W_i)$.

此外,算法的第 10 行中的条件 $D(W_i)>D(g_i)$是对算法的一种优化,没有这个条件,算法一样可以针对当前的目标集合进行回溯.然而,当 $D(W_i)\le D(g_i)$时可以看出:无论是对于算法第 11 行的变量 $D(W_i)$,还是第 15 行中 W_i 的输入 $D(g_k)$,其值都不可能有进一步的更新;而由于 $D(W_i)$不是无穷大,说明该 Web Services 的目标距离在以前的循环中曾经得到过更新,故而第 13 行的集合并操作已经在以前的循环中执行过.多次执行集合的并操作是没有意义的,而由于 $D(g_k)$没有任何变化,所以执行第 16 行,将其加入目标集合 S_g,自然不可能引起新的目标距离的变化.因此,执行第 16 行也是没有意义的.

该优化的现实意义可以这样解释:当已经得到了一个 Web Services 目标距离较小的估计之后,对于必然会增加其目标距离的估计,就不必再做了.

3 算法实现和对比分析

在上述算法的基础上,我们实现了一个 Web Services 自动组合和调用的原型系统 GOOSE(goal oriented

service engine).系统的体系结构如图 4 所示.

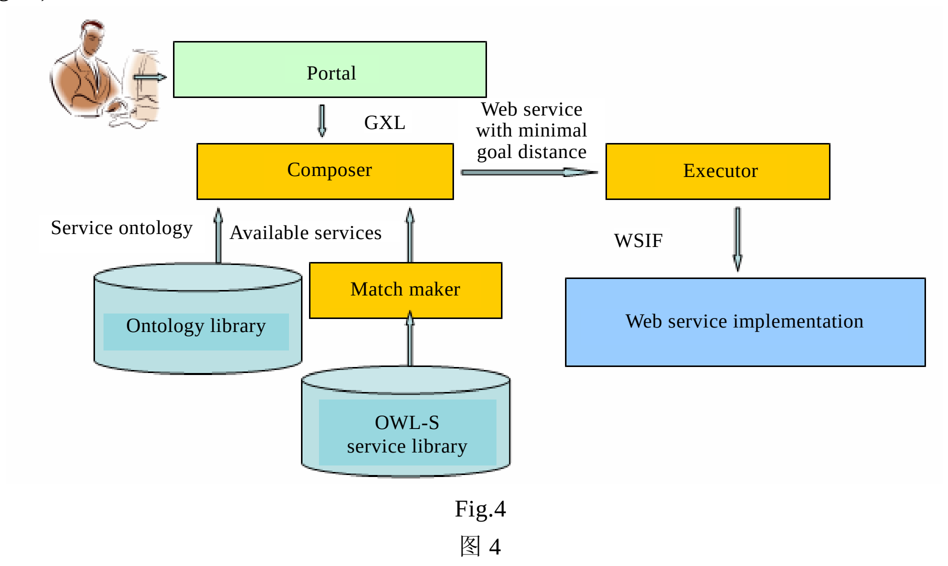


Fig.4
图 4

对于这个体系结构作如下说明:首先,用户通过一个客户界面提出用户的需求,这个需求通过一个形式化定义的 GXL(goal expression language)语言加以表示.而 GOOSE 系统的核心算法模块 Composer 将这个需求通过领域本体库和 Web Services 匹配器 Match Maker 去寻找可用的 Web Services,这些 Web Services 都是通过 OWL-S 语言来描述的;然后,Composer 通过上述的算法对这些 Web Services 寻找目标距离最小的 Web Services;最后,将找出来的 Web Services 交给执行模块 Executor 执行,同时接受 Executor 模块的反馈.而 Executor 在执行的时候,则通过 WSIF API[11]调用到真正的 Web Services 实现.

下面通过一个具体的业务例子来说明通过本文中的启发式 Web Services 组合算法,实际的 Web Services 在 GOOSE 系统中的组合过程.

这个业务的例子是用户需要了解著作《Effective Java》的作者是谁.用户将这个需求通过 GXL 语言提交给 GOOSE 系统.GXL 语言对于这个需求的表述如下:

```
import “http://owl.goose/book.owl#”;
provide
    BookName=“Effective Java”;
require
    Author ;
```

import 语句表示用户使用的本体库的名称;provide 语句表示的是用户提供的条件,书籍名称;require 语句表示用户需要的是这本书的作者.这个需求的表达非常类似于简单的 SQL 文法,界面如图 5 所示.

GOOSE 系统发现现有 3 个 Web Services:一个 Web Service 叫做 BookName2Author,可以通过一本书的名称得到作者;第 2 个 Web Service 是 BookName2ISBN,可以通过一本书的名称得到书籍的 ISBN 号码;最后一个 Web Service 是 ISBN2Author,可以通过书籍的 ISBN 号得到作者的名称.在这里,用户的需求可以通过两种方式来满足:第 1 种是直接调用 BookName2Author 这个 Web Service,从书籍的名称得到作者名;第 2 种方式是组合 BookName2ISBN 和 ISBN2Author 这两个 Web Service,得到作者的名称.

系统刚开始启动的时候,由于所有的 Web Services 的目标距离因子都为 1,所以,通过启发式算法可以判断出,在当前可用的两个 Web Services 中,BookName2Author 的目标距离为 1,BookName2ISBN 的目标距离为 2.所以,系统调用 BookName2Author,试图得到书籍的作者名.

假如因为网络等原因导致调用 BookName2Author 失败,GOOSE 系统将调节 BookName2Author 这个 Web Services 的目标距离因子为 1.5(δ(BookName2Author)=δ(BookName2Author)+1/2^1),然后调用 BookName2ISBN

得到书籍的 ISBN 号码.

在得到 ISBN 号码之后,当前 3 个 Web Service 都可用,而 BookName2Author 的目标距离为 1.5.ISBN2Author 的目标距离因子为 1,BookName2ISBN 的目标距离为 1.5(δ(BookName2ISBN)+δ(ISBN2Author)).

通过比较 3 个目标距离,系统调用 ISBN2Author,得到书籍的作者 Joshua Bloch.

经过这次运行,3 个 Web Service 的目标距离因子都得到了提高或降低,假如不同的用户再次提出同样的需求,算法通过目标距离的判断,将不会调用网络情况不佳的 BookName2Author 这个 Web Service,而是调用 BookName2ISBN 和 ISBN2Author 两个 Web Service 完成任务.

图 6 显示了两个同样的用户需求在系统中的运行结果.其中:第 1 次调用了 3 个 Web Services, BookName2Author 抛出了异常;第 2 次直接调用了 BookName2ISBN 和 ISBN2Author 两个 Web Services.

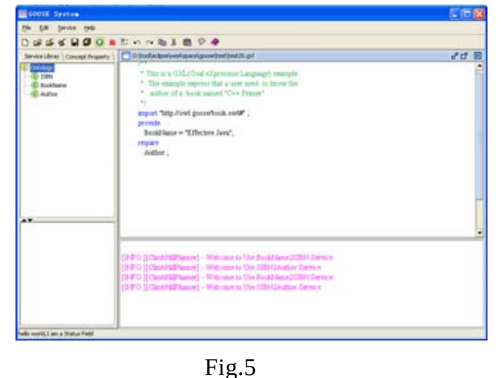


Fig.5

图 5

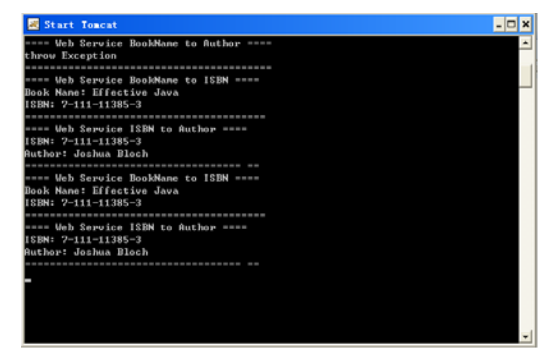


Fig.6

图 6

对于算法性能的评估采用将本算法和另外一种启发式的 Web Services 组合算法——BF*算法[12]以及简单地采用遍历的方式,对当前可行的 Web Services 进行逐个调用的算法进行了比较.实验环境部署在一台 CPU 为 AMD Athlon 1800+,内存为 512M 的 PC 机上,操作系统使用 Windows XP SP2,开发语言采用 Java,运行的 Java 虚拟机为 J2SE 1.5.0_04.鉴于目前没有相关的标准平台和标准测试数据集,实验数据采用随机生成的模拟 Web Services 以及随机生成的用户目标作为测试用例.实验分别选取了 500,750,1 000,1 250,1 500,1 750,2 000,2 250, 2 500 个 Web Services 作为组合的 Web Services 库.单个 Web Services 的执行时间选取为在本机上 30 个简单的 Web Services 执行时间的平均值.对于每个测试集合,我们分别测试 10 遍,然后计算其算术平均值和平均值的标准偏差(standard error of the mean),得到的实验结果如图 7 所示.

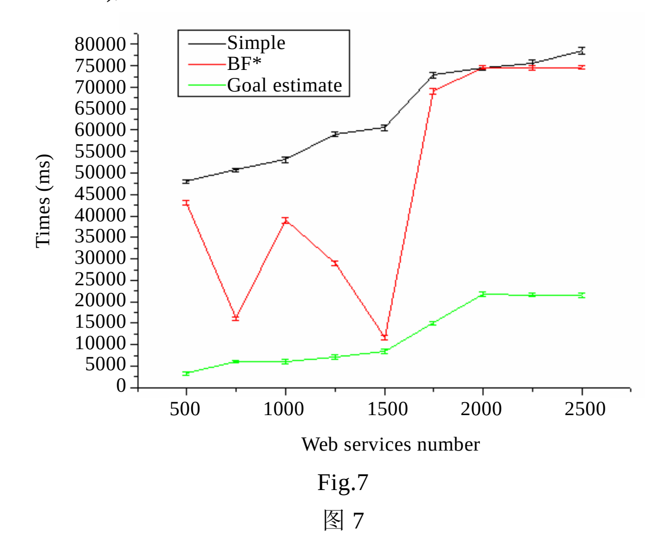


Fig.7

图 7

由图 7 可以看出:简单的遍历方法性能是最差的,而 BF*算法的性能变化随着测试集波动较大,这与 BF*算法的启发式规则有关.BF*算法认为:如果一个 Web Service 的输出集合与目标集合的交集较大,则优先选用这个 Web Service.这在有些测试集里面有非常好的性能;然而在有些测试集里面,其效率和简单的遍历方法差不多.本文提出的基于目标评估的算法,在各个测试集合里面则一直表现出较好的性能.

4 相关工作

除了在上文中提及的 BPEL4WS,EFlow 等手工 Web Services 组合方式以外,Web Services 的组合方式可以分成两种:半自动和全自动的组合方式.

半自动的组合方式往往需要专家手工编写一定量的领域模板,在用户提出需求以后,系统根据用户的需求选择某个模板.Stanford 大学是最早从事 Semantic Web Service 研究的大学,Sheila A. McIlraith 等人基于 Golog 语言的技术[13,14],实际上是一种领域模板;此外,胡海涛等人[15]提出的 Troll 方法,也是利用服务模板对多约束下的参数进行求解; Maryland 大学的 Dan Wu 等人[16]利用了成熟的 HTN 规划器 SHOP2 将 Web Service 的组合看作是领域任务的分解.本文使用了虚 Web Service 的概念,对需要专家干预的目标进行分解.其优点在于,这种对于特定领域的分解方案是可选的;缺点在于,虚 Web Service 没有对前提的各个目标实现顺序进行约束.

自动的 Web Services 组合方式往往将 Web Services 组合问题看作是一个状态空间的搜索问题,Proteus[17,18]系统就是一种基于前向式查找的自动组合系统,SWORD[19]系统则利用了基于规则的服务描述,使用专家系统来实现 Web Service 的组合.李曼等人[20]提出了一种基于领域本体关联度的 Web 服务动态组合算法,与本文类似的是,都是将 Web Service 看作是在一个图上的搜索,其核心思想是:在 Web 服务组合的过程中,优先选择在语义上最为相关的 Web Service;而本文提出的算法则是通过回溯的方式,在运行的过程中不断调整 Web Service 的目标距离因子,从而使 Web Service 的成功和失败影响后续 Web Service 的组合.Seog-Chan Oh[12]等人则提出了一种基于 A*算法的启发式算法.文中已经给出了详细的性能比较.

5 总结和进一步的工作

Web Service 的自动组合体现了以用户为中心的业务模式,用户的各种纷繁复杂的需求可以通过对 Web Service 的组合而不是重新编写代码来实现.本文提出了一种启发式的 Web Service 自动组合算法,用户只需提供目标状态、初始状态和约束条件就能够实现 Web Service 的组合,而系统在运行过程中还能够自动淘汰较差的 Web Service,本文中通过一个书籍的信息服务实例说明了这一点.

目前,这种组合方式还有待完善,因为在实际的业务例子中,用户的很多需求都是模糊的,并不能提供清晰的目标.在以后的工作中,我们将改善 GOOSE 系统,致力于让用户能够表达模糊的需求而不是明确的目标.

References:

[1] WSDL. Web services description language 1.1. W3C Note. 2001. http://www.w3.org /TR/wsdl

[2] SOAP. Simple object access protocol 1.2. W3C Recommendation. 2003. http://www.w3.org/TR/soap

[3] BPEL4WS Consortium. Business process execution language for Web services. 2003. http://www.ibm.com/Developerworks/library/ws-bpel

[4] Casati F, Ilnicki S, Jin LJ, Krishnamoorthy V, Shan MC. Adaptive and dynamic service composition in eFlow. In: Proc. of the 12th Int'l Conf. on Advanced Information Systems Engineering (CAiSE). 2000. http://www.hpl.hp.com/techreports/2000/HPL-2000-39.pdf

[5] Manola F, Miller E. RDF primer. 2004. http://www.w3.org/TR/REC-rdf-syntax/

[6] Smith MK, Welty C, McGuinness DL. OWL Web ontology guide. 2004. http://www.w3.org/TR/owl-guide/

[7] Berners-Lee T, Hendler J, Lassila O. The Semantic Web: A New Form of Web Content That is Meaningful to Computers Will Unleash a Revolution of New Possibilities. Scientific American Magazine, 2001,(5):35−43.

[8] Martin D, Burstein M, Hobbs J, Lassila O, McDermott D, Mcllraith S, Narayanan S, Paolucci M, Parsia B, Payne T, Sirin E, Srinivasan N, Sycara K. OWL-S: Semantic markup for Web services. 2004. http://www.daml.org/services/owl-s/1.1/overview/

[9] Erol K, Hendler J, Nau DS. UMCP: A sound and complete procedure for hierarchical task-network planning. In: Proc. of the Int'l Conf. on AI Planning Systems (AIPS). 1994. 249−254. http://www.cs.umd.edu/~nau/publications.html

[10] Erol K, Nau D, Hendler J. HTN planning: Complexity and expressivity. In: Proc. of the National Conf. on Artificial Intelligence. 1994. http://www.cs.umd.edu/~nau/publications.html

[11] Duftler MJ, Mukhi NK, Slominski A, Weerawarana S. Web services invocation framework. In: Proc. of the OOPSLA Workshop on Object-Oriented Web Services. 2001. http://www.extreme.indiana.edu/~aslom/papers/oopsla2001_workshop_wsif.pdf

[12] Oh SC, On BW, Larson EJ, Lee D. BF*: Web services discovery and composition as graph search problem. In: Proc. of the 2005 IEEE Int'l Conf. on e-Technology, e-Commerce and e-Service. 2005. http://ieeexplore.ieee.org/iel5/9634/30444/01402397.pdf

[13] McIlraith SA, Son TC, Zeng HL. Semantic Web services. IEEE Intelligent Systems (Special Issue on the Semantic Web), 2001,21(6):46−53.

[14] McIlraith S, Son TC. Adapting golog for programming the semantic Web. In: Proc. of the 5th Symp. on Logical Formalizations of Commonsense Reasoning. 2001. http://citeseer.ist.psu.edu/mcilraith01adapting.html

[15] Hu HT, Li G, Han YB. An approach to business-user-oriented large-granularity service composition. Chinese Journal of Computers, 2005,28(4):694−703 (in Chinese with English abstract).

[16] Wu D, Parsia B, Sirin E, Hendler J, Nau D. Automating DAML-S Web services composition using SHOP2. In: Proc. of the 2nd Int'l Semantic Web Conf. (ISWC 2003). Sanibel Island, 2003. http://www.cs.umd.edu/~nau/publications.html

[17] Ghandeharizadeh S, Knoblock CA, Papadopoulos C, Shahabi C, Alwagait E, Ambite JL, Cai M, Chen CC, Pol P, Schmidt R, Song SH,Thakkar S, Zhou RF. Proteus: A system for dynamically composing and intelligently executing Web services. In: Proc. of the 1st Int'l Conf. on Web Services (ICWS). Las Vegas, 2003. http://citeseer.ist.psu.edu/ghandeharizadeh03proteus.html

[18] Thakkar S, Knoblock CA, Ambite JL, Shahabi C. Dynamically composing Web services from on-line sources. In: Proc. of the AAAI-02 Workshop on Intelligent Service Integration. Edmondon, 2002. http://citeseer.ist.psu.edu/thakkar02dynamically.html

[19] Ponnekanti SR, Fox A. SWORD: A developer toolkit for Web service composition. In: Proc. of the Int'l World Wide Web Conf. Honolulu, 2002. 83−107. http://www2002.org/CDROM/alternate/786/

[20] Li M, Wang DZ, Du XY, Wang S. Dynamic composition of Web services based on domain ontology. Chinese Journal of Computers, 2005,28(4):644−650 (in Chinese with English abstract).

附中文参考文献:

[15] 胡海涛,李刚,韩燕波.一种面向业务用户的大粒度服务组合办法.计算机学报,2005,28(4):694−703.

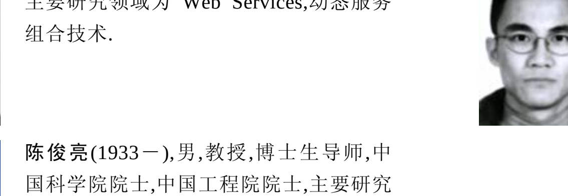
[20] 李曼,王大治,杜小勇,王珊.基于领域本体的 Web 服务动态组合.计算机学报,2005,28(4):644−650.



温嘉佳(1979－),男,广东揭西人,博士生,主要研究领域为 Web Services,动态服务组合技术.



彭泳(1978－),男,博士,讲师,主要研究领域为 Web 信息检索,Semantic Web,Web Services.





陈俊亮(1933－),男,教授,博士生导师,中国科学院院士,中国工程院院士,主要研究领域为网络智能化.